AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph beginning on page 10, line 7, as follows:

In Figure 7, instructions are shown in a sequence in which they are stored in memory. There is a first contiguous sequence of instructions 702 being executed by the processor, as indicated by the arrows, such as arrow 704, to the side of the contiguous set of instructions. Thus, the instruction 706 was first executed, as indicated by arrow 704, followed by instruction 708, as indicated by arrow 710. Various sets of instructions, such as a set of instructions 712, are repeatedly executed in a loop, as indicated by the backward-pointing arrows 714 and 716 in the case of the repeatedly executed set of instructions 712. In general, instructions are executed in order, except when a branch-type instruction directs execution to an instruction not following the branch instruction, such as a branch instruction 718 directing execution not to the subsequent instruction 720 but to the instruction 722, as indicated by arrow 724. Occasionally, for one of a variety of reasons, the normal execution thread, as determined by the order of instructions in memory and by the instructions themselves, such as branch instructions, is interrupted, as indicated by the dashed arrow ~~726~~ 727 in Figure 7. Instruction 726 executed and directed execution to instruction 728, as indicated by arrow 730. However, due to a trap arising from execution of instruction 726 or due to a fault or external interrupt arising from an attempt to execute instruction 728, the flow of instruction execution is interrupted, and instruction execution is directed to an interruption vector having, as its first instruction, instruction ~~730~~ 729. As shown in Figure 7, the interruption vector may contain a branch instruction 732 that directs subsequent instruction execution to an interruption handler routine, indicated in Figure 7 by a discrete set of contiguous instructions 734. Once the interruption is handled, the interruption handler executes a return from interrupt ("rfi") instruction 736 which restores the processor state to the state the processor was in when it was initially interrupted, and returns execution to the original flow of execution, as indicated by arrow 740 in Figure 7.

Please amend the paragraph beginning on page 11, line 3, as follows:

Figure 8 is a flow-control diagram that describes the steps taken by the processor to dispatch an interruption. When the interruption occurs, the processor, in step 802, stores the contents of the PSR and IP registers into the IPSR and IIP registers, stores the contents of the IP register at the time of execution of the last successfully executed instruction into the IIPA register, and updates additional interruption registers with new values reflective of the current interruption. In step 804, the processor updates the ISR register with information related to the type of interruption that has occurred, which, along with the identity of the interruption vector to which instruction execution has been directed, identifies the specific type of interruption that has occurred. In step 806, the processor updates the PSR register to place the processor into an interruption-handling state. As part of this update, the current priority level ("*cpl*") field within the processor status register, PSR.*cpl*, is updated to have the value "0", indicating the 0, or highest priority level. The Itanium processor handles all interruptions at the highest priority level. Next, in step 808, the processor places the address of an interruption vector (~~730~~ 729 in Figure 7) into the IP register and, in step 810, resumes execution by executing the first instruction in the interruption vector corresponding to the interruption pointed to by the contents of the IP register.